***Exception Handling***

An exception is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.

An exception can occur for many different reasons:

* A user has entered invalid data.
* A file that needs to be opened cannot be found.
* A network connection has been lost in the middle of communications
* the JVM has run out of memory.

Some of these exceptions are caused by user error, others by programmer error, and others by physical resources that have failed in some manner.

Types of Exception Handling:

1.Checked exception

2. Unchecked exception

1. Runtime exception

2. Error

***Checked***

* Exceptional conditions that a well-written application should anticipate and recover from

Example : File Reading process

* An application prompts a user for an input file name, then opens the file by passing the name.
* The user provides the name of an existing, readable file, and the execution of the application proceeds normally.
* If the user supplies the name of a nonexistent file an exception occurs
* A well-written program will catch this exception and

notify the user of the mistake.

***Unchecked***

***Error***

* Exceptional conditions that are external to the application, and that the application usually cannot anticipate or recover from
* Example : File processing
* An application successfully opens a file for input, but is unable to read the file because of a hardware or system malfunction.
* The unsuccessful read will throw Error.
* An application might choose to catch this exception, in order to notify the user of the problem
* But it makes sense for the program to print a stack trace and exit.

***Runtime Exception***

* These are exceptional conditions that are internal to the application, and that the application usually cannot anticipate or recover from.
* These usually indicate programming bugs, such as logic errors or improper use of an packages.
* Example : File Processing
* In the file reading application, if a logic error causes a null to be passed it will cause an Exception.
* The application can catch this exception, but it probably makes more sense to eliminate the bug that caused the exception to occur.

***How are Exceptions Handled?***

***Throwing an exception:***

* When an error occurs within a method, the method creates an object and hands it off to the runtime system – exception object
* The exception object contains information about the error
* Type of exception
* The state of the program when the error occurred
* Creating an exception object and handing it to the runtime system is called throwing an exception.

***Call Stack:***

* After a method throws an exception, the runtime system attempts to find something in the call stack to handle it.
* The list of methods that had been called to get to the method where the error occurred is known as the call stack.

***Catching the exception:***

* The search begins with the method in which the error occurred
* Proceeds through the call stack in the reverse order in which the methods were called.
* The runtime system passes the exception object to the appropriate “ exception handler - a method that contains a block of code that can handle the exception “
* The type of the exception object thrown should match the type that can be handled by the handler
* The exception handler chosen is said to catch the exception

***Static***

* The static variable can be used to refer the common property of all objects (that is not unique for each object)
* e.g. company name of employees,college name of students etc.
* The static variable gets memory only once in class area at the time of class loading.
* It makes the program memory efficient (i.e it saves memory).

***Final***

* Used to restrict the usage
* The final keyword can be used with

Variable

Method

class

* A method can be declared final if it has an implementation that should not be changed and it is critical to the consistent state of the object.
* A class that is declared final cannot be sub-
* If you make any method as final, you cannot override it. classed.
* During Inheritance , final method is inherited but you cannot override it

***Enum***

* An enum is a data type which contains fixed set of constants.
* It can be used for days of the week (SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY and SATURDAY) directions (NORTH, SOUTH, EAST and WEST) etc.
* Are static and final implicitly.
* Available from Java 5.
* Enums can be thought of as classes that have fixed set of constants.